D:\Logos\CG_logoReflect.tif

Description: C:\Users\244345\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\MobileTestingLogonew.png

**QE&A Technology CoE**

**7/13/2017**

C.R.A.F.T Cucumber

**Table of Contents**

[**1. Purpose** 3](#_Toc488763459)

[**2. Pre-requisite** 3](#_Toc488763460)

[**3. Desktop Automation Settings** 3](#_Toc488763461)

[**3.1. Working with Global Settings properties** 3](#_Toc488763462)

[**3.2. Working with TestNG** 3](#_Toc488763463)

[**4. Appium Automation Settings** 4](#_Toc488763464)

[**4.1. Working with Global Settings properties** 4](#_Toc488763465)

[**4.2. Working with TestNG** 4](#_Toc488763466)

[**5. Perfecto Automation Settings** 5](#_Toc488763467)

[**5.1. Working with Global Settings properties** 5](#_Toc488763468)

[**5.2. Working with TestNG** 5](#_Toc488763469)

[**6. Mint Automation Settings** 6](#_Toc488763470)

[**6.1. Working with Global Settings properties** 6](#_Toc488763471)

[**6.2. Working with TestNG** 6](#_Toc488763472)

[**7. SeeTest Automation Settings** 6](#_Toc488763473)

[**7.1. Working with Global Settings properties** 6](#_Toc488763474)

[**7.2. Working with TestNG** 7](#_Toc488763475)

[**8. Saucelabs Automation Settings** 7](#_Toc488763476)

[**8.1. Working with Global Settings properties** 7](#_Toc488763477)

[**8.2. Working with TestNG** 7](#_Toc488763478)

[**9. Parallel Execution and Listeners** 8](#_Toc488763479)

[**10. Working with Cucumber Runner Files** 8](#_Toc488763480)

[**11. Working with Step Definitions** 9](#_Toc488763481)

[**12. Leverage Reusable functions with Framework** 9](#_Toc488763482)

[**12.1. Working with Selenium, Appium, & Perfecto Reusable Functions** 9](#_Toc488763483)

[**12.2. Working with Broken Links** 10](#_Toc488763484)

[**13. Working with POM.xml** 11](#_Toc488763485)

[**14. Reporting Configuration** 11](#_Toc488763486)

[**14.1. Extended Cucumber Reports** 11](#_Toc488763487)

[**14.2. Allure Reporting** 12](#_Toc488763488)

[**15. Jenkins Integration with CRAFT Cucumber** 13](#_Toc488763489)

[**16. Execution flow of CRAFT Cucumber** 15](#_Toc488763490)

# **1. Purpose**

The purpose of this document is to provide description of Cucumber CRAFT Framework. This document also provides step by step instruction for implementing the advanced features.

# **2. Pre-requisite**

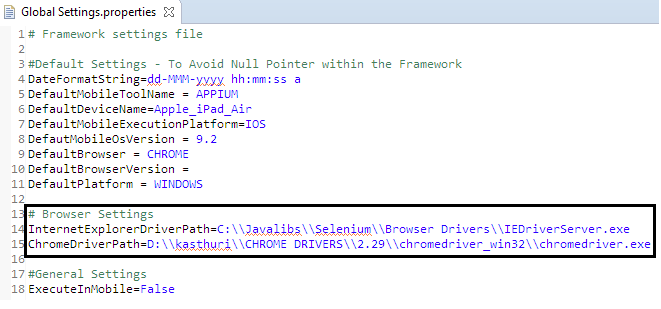
The user is expected to have basic knowledge on Automation Testing. Having a working knowledge of Cucumber Framework (Basic workflow), Jenkins and TestNG knowledge will be an added advantage.

* Install Cucumber plugin in Eclipse: http://cucumber.github.com/cucumber-eclipse/update-site

# **3. Desktop Automation Settings**

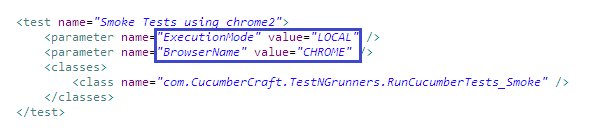
## **3.1. Working with Global Settings properties**

* Mention the respective InternetExplorerDriverPath/ ChromeDriver Path under Browser Settings
* Mention the ExecuteInMobile option as **False**



## **3.2. Working with TestNG**

* We have TestNG XML suite file in the framework under *src/test/resources* package.
* Configure testng.xml file with all required Test case, Test suite details.
* For Desktop Automation, set the following as,
  + **ExecutionMode** – LOCAL
  + **Browser** – CHROME/ FIREFOX/ SAFARI/INTERNET\_EXPLORER
* Pass the parameters and call the test classes to be executed as below,



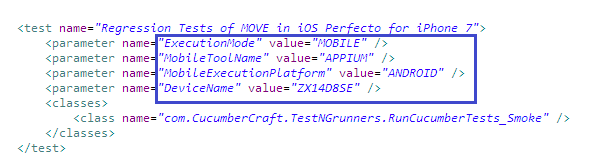
# **4. Appium Automation Settings**

## **4.1. Working with Global Settings properties**

* Mention **ExecuteInMobile** option as **True**
* Provide the Appium Settings in the property file as,
  + **AppiumURL**: Please provide the Appium running URL, generally it will be http://127.0.0.1:4723/wd/hub
  + **Application\_Package\_Name**: Provide the Package Name of Android Application
  + **Application\_MainActivity\_Name**: Provide the Main Activity Name of Android Application.
  + **iPhoneBundleID**: Provide the iOS application bundle ID.

## **4.2. Working with TestNG**

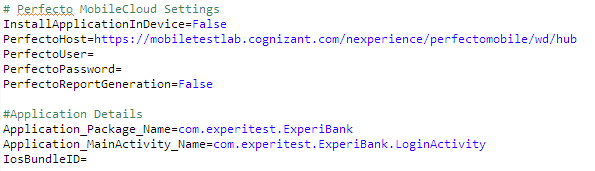
* Configure testng.xml file with all required Test case, Test suite details.
* For Appium automation, do the following
  + **ExecutionMode** – MOBILE
  + **MobileToolName** – APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



# **5. Perfecto Automation Settings**

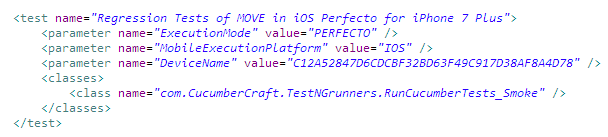
## **5.1. Working with Global Settings properties**

* Mention **ExecuteInMobile** option as **True**
* Provide the Perfecto Automation Settings in the property file as,
* **PerfectoHost** – Mention the Host details of Perfecto Mobile Cloud as to access the Cloud devices
* **PerfectoUser, PerfectoPassword** – Mention the authentication details of Perfecto Mobile Cloud
* **PerfectoReportGeneration** – Mention True or false to enable or disable the report generation in Craft local Reports folder
* **Application\_Package\_Name**: Provide the Package Name of Android Application
* **Application\_MainActivity\_Name**: Provide the Main Activity Name of Android Application.
* **iPhoneBundleID**: Provide the iOS application bundle ID.



## **5.2. Working with TestNG**

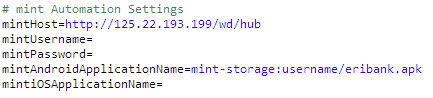
* For Perfecto Automation, provide the following data
  + **ExecutionMode –** Perfecto
  + **MobileExecutionPlatform** -ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** -Provide the respective device serial name or UDID/ **ManufacturerName , ModelName**



# **6. Mint Automation Settings**

## **6.1. Working with Global Settings properties**

* Mention **ExecuteInMobile** option as **True**
* Provide the Mint Automation Settings in the property file as,
* **mintHost** - Mention the Host details of mint to access the Cloud devices
  + **mintUsername, mintPassword** - Provide the authentication details of mint Cloud
  + **mintAndroidApplicationName** - Mention the Android Application Name to specify the Native or Hybrid Application under test
  + **mintiOSApplicationName** - Mention the iOS Hybrid or Native application name which is under test.



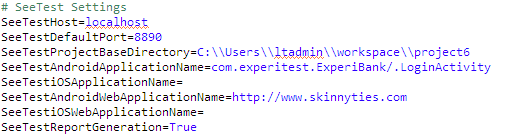
## **6.2. Working with TestNG**

* For mint automation, do the following
  + **ExecutionMode** – MINT
  + **MobileToolName** – APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID

# **7. SeeTest Automation Settings**

## **7.1. Working with Global Settings properties**

* Mention **ExecuteInMobile** option as **True**
* Provide the SeeTest Automation Settings in the property file as,
* **SeeTestHost** - This property needs to be set “localhost”.
* **SeeTestDefaultPort** - By default SeeTest provide “8890”,”8889”. Any one port number can be used in SeeTestDefaultPort.
* **SeeTestProjectBaseDirectory** - This directory is used by SeeTest Automation for storing image objects. We need to give full path with directory name.
* **SeeTestAndroidApplicationName** – Provide Android application name if the application under test is Native or Hybrid
* **SeeTestiOSApplicationName** - Provide iOS application name if the application under test is Native or Hybrid **SeeTestAndroidWebApplicationName** - Mention Web Application name for Android.
* **SeeTestiOSWebApplicationName** - Mention Web Application name for iOS.
* **SeeTestReportGeneration** – Mention true to generate SeeTest Automation reports along with C.R.A.F.T.



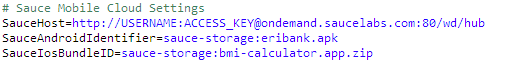
## **7.2. Working with TestNG**

* Configure testng.xml file with all required Test case, Test suite details.
* For SeeTest Automation, do the following
  + **ExecutionMode –** Perfecto
  + **MobileExecutionPlatform** -ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** -Provide the respective device serial name or UDID

# **8. Saucelabs Automation Settings**

## **8.1. Working with Global Settings properties**

* Mention **ExecuteInMobile** option as **True**
* Provide Saucelabs Host, Android or iOS application details in the property file as,

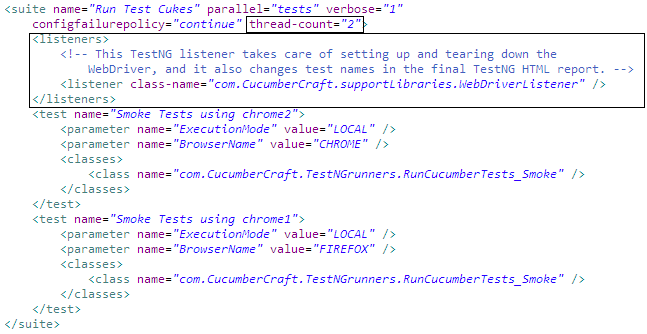


## **8.2. Working with TestNG**

* Configure testng.xml file with all required Test case, Test suite details.
* For Saucelabs test configuration, do the following
  + **ExecutionMode** – SAUCELABS
  + **MobileToolName** – APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID

# **9. Parallel Execution and Listeners**

* As per the TestNG Concepts, we can achieve Parallel execution by increasing the number of thread-count, it will ideally invoke Test methods in parallel.
* In Cucumber CRAFT, TestNG suites are used for batch/parallel execution. Every test method here will execute Cucumber Runner file which can trigger multiple features.
* We also have Listeners which help us to get the Parameters values from Test Method. This Listener will be called before Test Method invocation. Below is the sample,



# **10. Working with Cucumber Runner Files**

* The Runner files are responsible to invoke feature files accordingly.
* Cucumber Runner files are TestNG files which are present under /src/test/java/ -> com.CucumberCraft.TestNGrunners package
* Configure the Test runner in @CucumberOptions annotation.
  + **features** - Path of Feature files
  + **glue** - Path of the glue code/Step Definitions
  + **tags** - Mention the tags which are used as a filter to pick out specific scenarios. We can even have multiple tags.



# **11. Working with Step Definitions**

* Every Step Definition should extends to MasterStepDefs



* Driver initializations are handled in DriverManager.java file.
* Get the respective drivers using,

|  |  |
| --- | --- |
| Execution Mode | Driver initialization |
| Desktop Selenium |  |
| Appium, Perfecto, Saucelabs, mint |  |
| SeeTest |  |

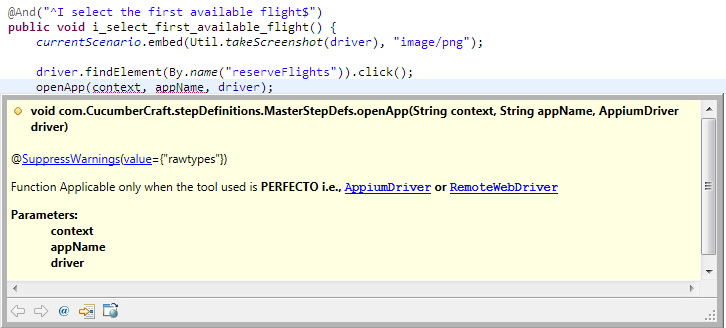
* We also have CukeHooks.java which has main driver creation logic and quitting logic.
* Make sure both the files, MasterStepDefs.java and CukeHooks.java are in same package of stepDefinition files.

# **12. Leverage Reusable functions with Framework**

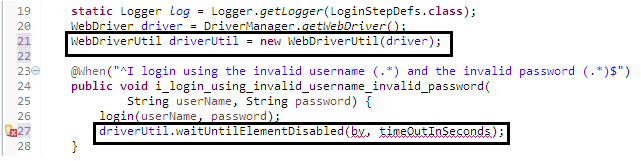
* Framework provides reusable functions which can be called directly into your Step Definitions. All Reusable functions are available in MasterStepDefs.java.

## **12.1. Working with Selenium, Appium, & Perfecto Reusable Functions**

* One can directly access resuable functions from Step Definitions.
* All details like which ExectionMode, Parameters has been mentioned for ease of use.

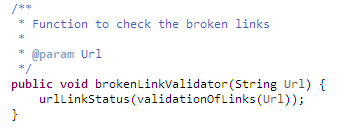


* We have also class called WebDriverUtil.java, this has commonly used selenium reusable functions, and this can be accessed by initializing WebDriverUtil in any Step Defintion java file.



## **12.2. Working with Broken Links**

* Framework provides reusable functions to get the broken link response.
* One can directly access broken link reusable function from Business Component.



# **13. Working with POM.xml**

* POM.xml has all the dependencies, plugins and other artifacts that are required for executing the Cucumber CRAFT project
* Maven-Surefire-plugin is used for executing TestNG suites.
* We can create multiple profiles in maven which can trigger an TestNG suite xml file, below is the sample



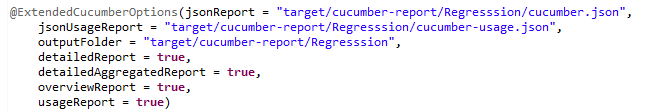
* Execute the maven POM using maven commands
* Run the profiles defined in the POM file, which has TestNG suites,

**mvn clean test -P ProfileName**

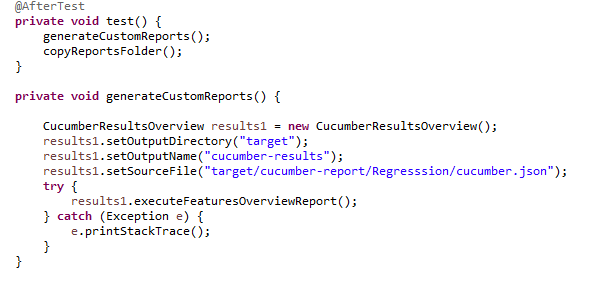
# **14. Reporting Configuration**

## **14.1. Extended Cucumber Reports**

* We have used Extended Cucumber Option to get the Overview and Detailed report.
* Please provide the appropriate path in Runner file accordingly as below,



* Though we have given the path, we would need to consolidate and generate report from basic cucumber.json, below is the sample code for generating additional reports. We have called this @AfterTest



## **14.2. Allure Reporting**

* Allure provides a clear 'big picture' of what features have been covered, where defects are clustered, how the timeline of execution looks like and many other convenient things.
* It gives Cumulative execution report (i.e., Overall report statistics)
* Allure reporting plugins and dependencies are mentioned in the POM.xml file

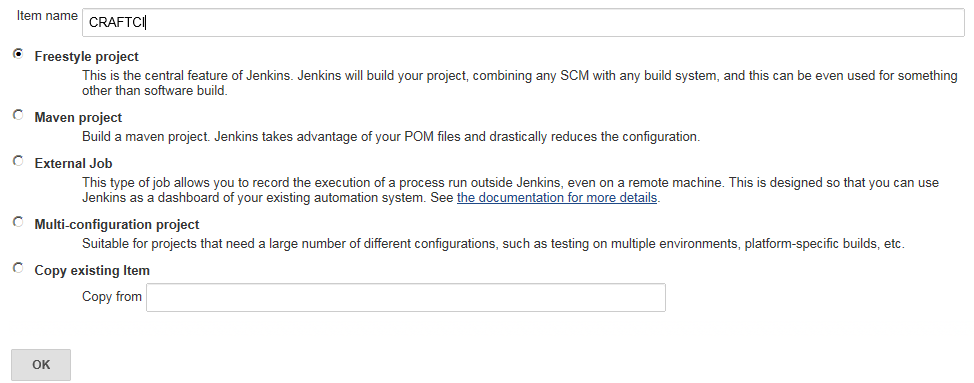


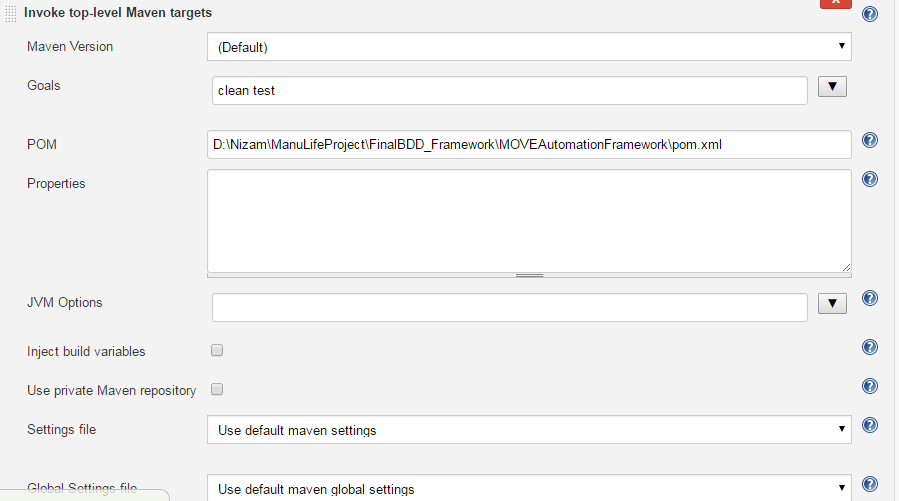
* Execute the POM.xml file using maven commands to generate Allure reports

**mvn test -P ProfileName site**

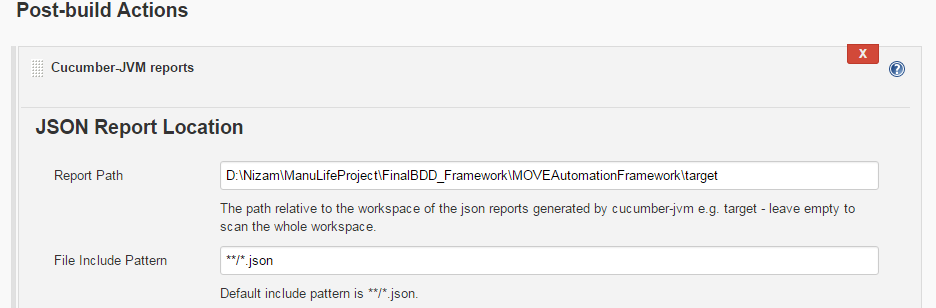
* Reports are generated under target/ folder.

# **15. Jenkins Integration with CRAFT Cucumber**

* Please download the Jenkins and configure accordingly.
* Now add a new job and call it CRAFT\_Cucumber, select Build a free-style software project.  
  [jenkins-simple-ant-build-project-4](http://eureka.ykyuen.info/?attachment_id=14680)
* After the job is created, configure the build. In Build section, select Invoke top-level Maven targets.



* Post build, we can achieve Cucumber Report in Jenkins, please download the **Cucumber-report plugin** for Jenkins



# **16. Execution flow of CRAFT Cucumber**

* Basic execution flow of CRAFT Cucumber is,
  + Execution starts from POM file
  + POM has TestNG suite configuration which will get triggered.
  + TestNG suites will contain the set of test cases (Cucumber Runner tests)
  + Now, Runner file will trigger the Cucumber features and Step definitions which are configured
  + Finally, Reports (Extended reports & Allure reports) are generated in the respective folders.